Notice of Allowability	Application No.	Applicant(s)	
	10/003,376	TOBIN, JOHN P.E.	
	Examiner	Art Unit	
	Tuan A. Vu	2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *2/01/07*.

2. ☒ The allowed claim(s) is/are *16-27, 29-41, 43-54, and 69-70(renum 1-39)* .

3. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
   a) ☐ All b) ☐ Some* c) ☐ None of the:
      1. ☐ Certified copies of the priority documents have been received.
      2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
      3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

4. ☐ A SUBSTITUTE OATH OR DECLARATION must be submitted. Note the attached EXAMINER'S AMENDMENT or NOTICE OF INFORMAL PATENT APPLICATION (PTO-152) which gives reason(s) why the oath or declaration is deficient.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   (a) ☐ including changes required by the Notice of Draftsperson's Patent Drawing Review (PTO-948) attached
      1) ☐ hereto or 2) ☐ to Paper No./Mail Date _____.
   (b) ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).

6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☒ Notice of References Cited (PTO-892)
2. ☐ Notice of Draftperson's Patent Drawing Review (PTO-948)
3. ☐ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
4. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
5. ☐ Notice of Informal Patent Application
6. ☒ Interview Summary (PTO-413), Paper No./Mail Date *2/23/07* .
7. ☒ Examiner's Amendment/Comment
8. ☒ Examiner's Statement of Reasons for Allowance
9. ☐ Other _____.

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 2/1/2007.

As indicated in Applicant's response, claims 16-17, 19, 22, 26-27, 30, 32-37, 39-40, 44-51, 53-54, 69 have been amended, claims 1-15, 55-68 canceled, and claim 70 added. Claims 16-54, 69-70 are pending in the office action.

EXAMINER'S AMENDMENT

2. An examiner's amendment to the record appears below. Should the changes and/or additions be unacceptable to applicant, an amendment may be filed as provided by 37 CFR 1.312. To ensure consideration of such an amendment, it MUST be submitted no later than the payment of the issue fee.

Authorization for this examiner's amendment was given in a telephone interview with Att. Steve Sullivan, Reg #38329 on 2/20/07.

The application has been amended as follows.

In the CLAIMS:

Claim 16:

A method for obfuscation of computer program execution flow to increase computer program security, the method comprising:

(a) partitioning the computer program into one or more non-critical code segments and one or more critical code segments to be obfuscated;

(b) removing at least one of the critical code segments and embedding the critical code segments within one or more exception handlers;

(c) providing the computer program with an exception set-up handler to set up operation of the exception handlers during the execution of the computer program;

(d) providing the computer program with ~~an~~ one or more in-line code segments inserted within at least one of the non-critical code segments ~~for setting up and invoking the exception set handler~~, wherein each of the in-line code segments forces an exception when executed and wherein the in-line code segments are placed in the computer program at points where the critical segments need to execute in order to maintain execution flow of the computer program; and

(e) executing the computer program on a computer having an operating system that provides kernel-level and user-level execution modes, and debug resources to support generation and processing of exceptions at specified addresses, wherein execution flow of computer program is maintained with the non-critical code segments executing in the user-level ~~protected~~ execution mode, but ~~when an address immediately prior to each one of the non-critical code segments in the execution flow is encountered~~ when one of the in-line code segments is executed, an exception ~~occurs~~ is forced and the kernel-level execution mode is entered whereby control is transferred to the one or more of the exception handlers previously set-up by the exception set-up handler for execution of the critical code segments, and when the exception handlers complete execution, control is returned to non-critical code segments, whereby program execution

flow is obfuscated from other programs running in user-level execution mode, thereby increasing difficulty of reverse engineering the computer program.

Claim 18:

The method of claim 16 wherein the ~~embedding~~ providing (d) further includes: executing the in-line code segment prior to a point in the program flow where any of the critical code segments was removed for placement into the exception handlers.

Claim 21:

The method of claim 17 wherein the ~~embedding~~ providing (b) further includes: inserting each of the exception handlers into the linked list.

Claim 28: (Cancel)

Claim 31:

The method of claim 30 further including rearranging non-~~embedded~~ critical code segments out of normal execution order, and setting exception addresses and return addresses to ensure proper program sequencing to further obfuscate the program execution flow.

Claim 36:

A method for obfuscation of computer program execution flow to increase computer program security, the method comprising:

(a) partitioning the program into one or more non-critical code segments and one or more critical code segments;

(b) obfuscating at least one of the critical code segments by removing the critical code segments and embedding each of one of the critical code segments within one or more exception handlers;

(c) providing the computer program with a driver to set up operation of the exception handlers during execution of the computer program; ~~and~~

(d) providing the computer program with ~~an~~ one or more in-line code segments inserted within ~~a~~ at least one of the non-critical code segments ~~for invoking the driver when the program is executed~~, wherein each of the in-line code segments forces an exception when executed and wherein the in-line code segments are placed in the computer program at points where the critical segments need to execute in order to maintain execution flow of the computer program; and

(e) executing the computer program on a computer having an operating system that provides kernel-level and user-level execution modes, and debug resources to support generation and processing of exceptions at specified addresses, wherein execution flow of computer program is maintained with the non-critical code segments executing in the user-level ~~protected~~ execution mode, but ~~when an address immediately prior to each one of the non-critical code segments in the execution flow is encountered~~ when one of the in-line code segments is executed, an exception ~~occurs~~ is forced and the kernel-level execution mode is entered whereby control is transferred to the one or more of the exception handlers previously set-up by the driver for execution of the critical code segments, and when the exception handlers complete execution, control is returned to non-critical code segments, whereby program execution flow is obfuscated from other programs running in user-level execution mode, thereby increasing difficulty of reverse engineering the computer program.

Claim 42: (Cancel)

Claim 43:

The method of claim ~~42~~ 41 further including: modifying debug registers during execution of the exception handlers such that any number of exception handlers may be daisy chained.

Claim 47:

The method of claim ~~42~~ 41, further comprising providing code for the exception handlers to modify the return address to be used after completion of the exception processing.

Claim 49:

The method of claim ~~42~~ 41, further comprising including code within the exception handlers to modify exception conditions to support future exceptions.

Claim 53:

The method of claim ~~42~~ 41, further comprising embedding one or more exceptions within one or more other exception handlers, to further obfuscate the program execution flow.

Claim 69:

A system for obfuscation of program execution flow, comprising:

a computer system including a processor, memory, an operating system that provides kernel-level and user-level execution modes, and debug resources to support the generation and processing of exceptions at specified addresses; and

an obfuscated program comprising;

one or more non-critical code segments,

a plurality of code[s] segments determined to be critical code segments that have been removed and ~~encapsulated~~ embedded within one or more exception

handlers, wherein the exception handlers are provided as part of the obfuscated program, ~~and~~

one or more in-line code segments inserted within at least one of the non-critical code segments, wherein each of the in-line code segments forces an exception when executed and wherein the in-line code segments are placed in the obfuscated program at points where the critical segments need to execute in order to maintain execution flow of the computer program, and

a set-up handler for setting up execution of the exception handlers by setting up the debug resources, ~~such that~~

wherein when the computer system executes the program, execution flow of program is maintained with the non-critical code segments executing in the user-level ~~protected~~ execution mode, but when ~~an address immediately prior to each one of the non-critical code segments in the execution flow~~ one of the in-line code segments is ~~encountered~~ executed, an exception ~~occurs~~ is forced and the kernel-level execution mode is entered whereby control is transferred to the one or more of the exception handlers previously set-up by the set-up handler for execution of the critical code segments, and when the exception handlers complete execution, control is returned to non-critical code segments, whereby program execution flow is obfuscated from other programs running in user-level execution mode, thereby increasing difficulty of reverse engineering the computer program.

Claim 70:

A method for obfuscation of computer program execution flow to increase computer program security, the method comprising:

(a) partitioning the computer program into one or more non-critical code segments, and one or more critical code segments to be obfuscated;

(b) removing from the computer program, at least one of the critical code segments and embedding the removed critical code segments within one or more exception handlers, wherein the exception handlers run under kernel-level execution mode of a computer's operating system;

(c) adding to the computer program, the exception handlers and code for setting-up the exception handlers during the execution of the program;

(d) adding to the computer program, code for ~~invoking at least one of the exception handlers~~ forcing an exception when executed at each location in the computer program where the critical code segments were to be invoked in the computer program prior to removal; and

(e) executing the computer program on the computer having an operating system that provides kernel-level and user-level execution modes, wherein execution flow of computer program is maintained where the non-critical code segments execute in the user-level ~~protected~~ execution mode, and ~~where~~ when each location in the computer program previously occupied by the critical code segments is encountered, an exception ~~occurs~~ is forced by execution of the code previously added for the exception-forcing and the kernel-level execution mode is entered whereby control is transferred to the one or more exception handlers previously set up by the code for setting up the exception handlers, for execution of the critical code segments, and when the exception handlers complete execution, control is returned to the non-critical code segments, whereby program execution flow is obfuscated from other programs running in user-level execution mode, thereby increasing difficulty of reverse engineering the computer program.

EXAMINER'S STATEMENT OF REASONS FOR ALLOWANCE

3. Claims 16-27, 29-41, 43-54, and 69-70 are allowed.

The following is an examiner's statement of reasons for allowance.

The prior art taken separately or jointly does not suggest or teach the following features.

A method for obfuscating program execution flow, the program comprising (i) non-critical code segments being partitioned from critical code segments, the critical code segments destined to be obfuscated and being removed from the computer program by way of embedding in place of the removed critical code segments one or more exception handlers; (ii) providing a set-up code within the computer program for setting up the exception handlers and inserting code within the non-critical segments to force an exception so to invoke the handlers; (iii) executing the computer program in a operating system provided with kernel-level and user-level execution mode, wherein the execution of the computer program is maintained within execution of the user-level mode of the non-critical code segments; such that upon encountering a location of any removed critical code segment, the inserted code previously inserted in the non-critical code segments forces an interrupt, and control is transferred to the exception handlers previously set by the set-up code; thereby the kernel-level code execution is entered for execution of the critical code segments; and (iv) upon completion of said handlers, control is transferred back to the non-critical code segments, enabling program execution flow to be maintained while obfuscating all critical parts of the program from other programs running in user-level mode, thereby increasing difficulty of reverse engineering the computer program; as recited in claims 1, 16, 36, 69, and 70.

Kuraza, USPN: 5,450,586, teaches inserting using some setup code to install the markers at suspicious areas of code, markers indicative of task switch during a debug program execution,

enabling via using the underlying non-intrusive code triggered by the markers, to perform low level operations and providing the user with trace information, and based on an interface via which user's analysis and intervention, enabling further tracing, rearranging the code marker maintenance for subsequent optimizing, tracing and recompiling and improving the code. Kuzara's code marker insertion does not provide or suggest the obfuscating scheme by removing critical code from a user-level execution program and replacing each of which with pre-inserted instructions at the very location thereof that forces a runtime exception at such location, thus enabling said critical code to be executed in a handler in a kernel level without stopping the flow of the program which is designed to prevent reverse engineering as set forth in the sequence of (i) to (iv) above.

Mealey et al, USPN: 5,790,846, teaches set up a table for registering handler information which can be referred to when a kernel handler is invoked by way of an exception or interrupt event; but does not teach or suggest the combination of steps (i) to (iv) to obfuscate some critical code while maintaining the flow of the main user-level program.

Delong, USPN: 6,247,169, teaches embedded exception handlers with layers of handlers wherein data resulted from a higher layer handlers enable lower handler to be selectively called to provide fault information thus obviate the user with the need for further intervening action. The embedded scheme by Delong remains a debug paradigm to handle standard exception events and does not teach or suggest the removing of (i) and replacing of (ii), and does not teach maintaining of user-level execution via implementing code insertion and critical-code execution as in (iii) and (iv).

Any comments considered necessary by applicant must be submitted no later than the payment of the issue fee and, to avoid processing delays, should preferably accompany the issue fee. Such submissions should be clearly labeled "Comments on Statement of Reasons for Allowance."

Conclusion

4. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng-Ai An can be reached on (571)272-3756.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free)

Tuan A Vu
Patent Examiner,
Art Unit 2193
February 23, 2007